

# VRFP: On-the-fly Video Retrieval using Web Images and Fast Fisher Vector Products

Xintong Han\*, Bharat Singh\*, Vlad I. Morariu, Larry S. Davis

Center for Automation Research, University of Maryland, College Park  
{xintong, bharat, morariu, lsd}@umiacs.umd.edu

## ABSTRACT

VRFP is a real-time video retrieval framework based on short text input queries in which weakly labeled training samples from the web are obtained, after the query is known. Our experiments show that a Fisher Vector is robust to noise present in web-images and compares favorably in terms of accuracy to other standard representations. While a Fisher Vector for a new query can be constructed efficiently, matching against the test set is slow due to its high dimensionality. To perform matching in real-time, we present a lossless algorithm for accelerating the computation of dot product between high dimensional Fisher Vectors. We prove that the expected number of multiplications required is quadratic in terms of sparsity in Fisher Vectors. We are not only able to construct and apply query models in real-time, but with the help of a simple re-ranking scheme, we also outperform state-of-the-art automatic retrieval methods by a significant margin on TRECVID MED13 (3.5%), MED14 (1.3%) and CCV datasets (5.2%).

## Keywords

Video Retrieval; Web-Based Retrieval; Fisher Vectors; Fast dot products

## 1. INTRODUCTION

With the reducing cost of cloud storage like iCloud, Dropbox, Google Drive, etc., people have started collecting and storing personal videos on a large scale. A typical user's media library may contain thousands of videos after a few years. However, videos in such libraries do not have labels and it eventually becomes very tedious to search them. Therefore, we would like to develop a search engine - which can perform visual search in a user's personal media library efficiently. Like any search engine, input should only be a short text query, rather than a detailed description.

Retrieving relevant videos using only a *text query* is challenging because no training samples are available. One approach to this problem is to train a large concept bank (a set of pre-trained detectors) and apply them to database videos as a pre-processing step [40, 6, 29]. A distance measure (relevance) is computed between a query and those pre-trained concepts by combining the responses of the pre-trained concept detectors to the query weighted by their relevance (semantic similarity to the query). A motivation for having pre-trained concept detectors is that they can be run a priori when a human adds a video (i.e., during an indexing step), and when a new query is provided, retrieval can be performed

in real-time. However, designing and implementing this so called 'zero-shot' video retrieval pipeline requires significant engineering effort. Current methods select pre-trained concepts (which may be object/action/scene detectors) from multiple strongly annotated datasets like ImageNet [8], UCF101 [36], Sports 1 million [26], or from multiple modalities like OCR and speech recognition.

However, it is not clear which pre-defined concepts will be relevant to some arbitrary query. Since current methods rely on strongly annotated datasets [29, 28, 43, 23] (which are very small compared to weakly labeled data on the web), they would struggle to retrieve relevant results for queries such as 'Inside view of Notre-Dame', or 'Christmas near Rockefeller Center' if *inside view of Notre-Dame* or *Rockefeller Center* were not in the set of pre-trained concepts.

Most of these 'zero-shot' methods also assume that the query is a rather lengthy and detailed textual description [4, 40, 7, 23]. The event kit description in TRECVID for a query like "birthday party" contains information including what defines a "birthday party", common scenes or objects found in a birthday party, popular activities performed and even typical audio clips. Such a detailed textual description would rarely be entered by users querying a search engine.

Instead of using pre-defined concept bank approaches, recent work has also focused on building concepts only after the query is given. These methods [4, 34] crawl the web to discover concepts relevant to the query. Finally, "visualness" verification or concept pruning is performed and concept detectors are applied to database videos to obtain a final ranking. Since training data is collected only after the query is given, the semantic gap between the query and training images is much smaller than it is for pre-trained concepts. Moreover, such methods do not rely on strongly annotated datasets as they automatically obtain weakly annotated data from the web. Although this process is completely automatic, discovering *multiple* concepts given the query, downloading images related to hundreds of concepts and then training and applying detectors on every database video frame takes significant time. Thus, these methods are not as efficient as pre-trained concept based methods, so they cannot be used for real-time video retrieval.

Therefore, in contrast to discovering multiple concepts related to a query, on-the-fly retrieval approaches [2, 3, 13, 1] obtain training samples from the web only for the text-query (instead of mining similar concepts). However, web-images come with noisy samples, so it is important to have an image collection representation which is robust to noise. Some methods improve performance by using different images as separate queries [1], while other on-the-fly approaches focus on outlier removal algorithms [2] rather than designing a robust representation. Such on-the-fly approaches have only been demonstrated to work well for object/scene retrieval in images [2, 3, 13, 1] and face retrieval in videos [2]. To the best of

\*The first two authors contributed equally to this paper.

our knowledge, there is no existing work which performs on-the-fly retrieval for complex queries in videos. Our work involves the following novel contributions:

- We demonstrate that training a visual detector on-the-fly outperforms methods which use pre-trained concepts or discover concepts for complex event retrieval on three standard video retrieval datasets.
- We demonstrate that a simple dot product between Fisher Vectors constructed on CNN features of web-images obtained from the query and database video frames is robust to noise present in a web image collection and is an effective similarity measure between web images and videos.
- We show that Fisher Vectors constructed from video frames and web-images are sparse. Using sparsity and locality, we present a lossless algorithm to accelerate the computation of dot product between Fisher Vectors. Finally, we prove that on expectation, the number of arithmetic operations required by the algorithm is quadratic in terms of sparsity in Fisher Vectors.

## 2. RELATED WORK

Most video retrieval methods that do not assume knowledge of the query at training time (and thus do not train for specific queries) pre-train a large set of concept detectors. The corresponding detector confidences on dataset videos are then used to form semantic representations [40, 6, 16, 28]. Wu et al. [40] use off-the-shelf concept detectors and multimodal features to represent a video. The event description and video features are then projected to a high-dimensional semantic space. Finally, the event/video similarity scores are computed in this space to rank videos. By harvesting web videos and their descriptions, Habibian et al. [16] learn an embedding by jointly optimizing the semantic descriptiveness and the visual predictability of the embedding. Mazloom et al. [28] generate concept prototypes from a large web video collection, where each concept prototype contains a set of frames that are relevant to a semantic concept. The similarity between the event description and a video is measured by mapping the video frames in the concept prototype dictionary.

However, it is difficult to decide which concepts should be defined and trained without a priori knowledge of the query. If an event that needs to be retrieved has a large semantic gap between its description and the concept bank, methods that depend on a pre-defined concept bank will have poor performance. To reduce the semantic gap between the video description and the concept bank, some recent works discover the concepts after the event description is given [4, 34]. Chen et al. [4] query the verb-noun pairs in the event description in Flickr, and select visually meaningful concepts based on the tags associated with Flickr images. Then, 2,000 detectors are trained using web images and applied to dataset videos. Based on [4], Singh et al. [34] build pair-concepts and select the relevant concepts by a series of concept pruning schemes.

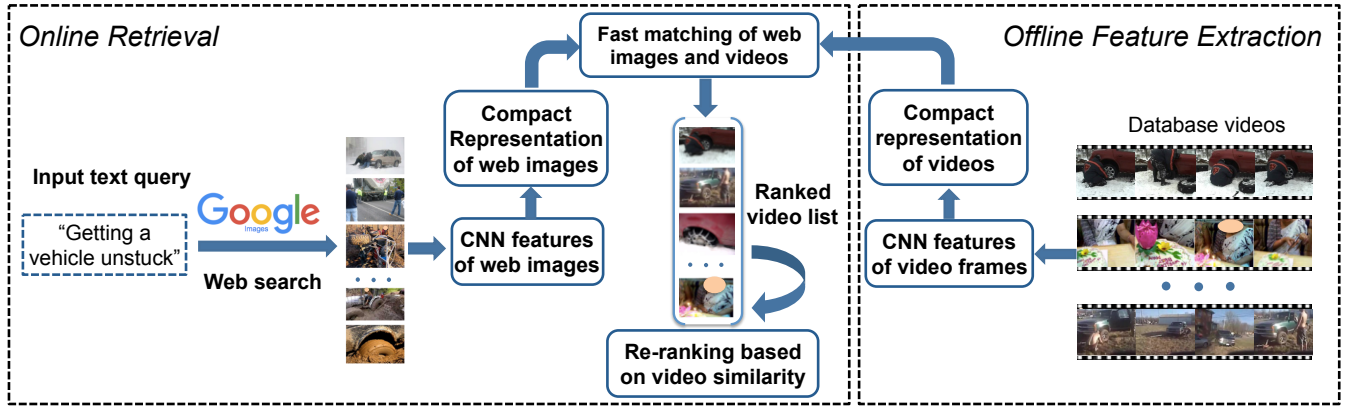
Although concepts used to detect the event of interest are semantically related (when they are obtained after the query is given), these event-driven concept detectors need to be applied to almost every frame in the test set, which makes this process computationally expensive [4, 34]. In contrast, on-the-fly retrieval methods do not have this problem, as they only collect web-images for the search query [2, 3, 13, 1]. On-the-fly methods have been used for large scale object retrieval [3, 2, 13], face retrieval in videos [2], and place and logo recognition [1]. As time has progressed, search engines have become more advanced. We show that now they can even be used for collecting training samples for complex

queries, especially, when no visual prior is available. However, the method used for similarity between web-images and videos can impact retrieval performance significantly. For on-the-fly retrieval, a linear SVM is typically trained [2, 3, 13, 1], as it is fast to train and predict. In this work, we show that measuring similarity using dot product between Fisher Vectors [31, 32] on CNN features performs significantly better than training a linear SVM on web images or computing dot product between average pooled CNN features.

Although Fisher Vectors are a good representation for video frames, they are high dimensional. Hence, computing their dot product is significantly more expensive (around 20-50 times) than applying a linear SVM on average pooled CNN features of video frames. In the past, several works have looked into accelerating retrieval using Fisher Vectors or other high dimensional features [19, 33, 18, 14]. One of the most useful techniques is product quantization (PQ) [17]. In this method, each feature vector is decomposed into equal length sub-vectors, and a lookup table is constructed at query time which helps us compute dot-products between two sub-vectors efficiently. To further accelerate PQ, a High Variance Subspace First approach was proposed [44]. In this method, instead of computing distances to all the codewords, only the distance from high variance codewords is computed to reduce computation. However, these methods take a hit in performance as they are lossy compression techniques. To speedup dot products without compromising accuracy, sparse matrix multiplication algorithms have been proposed [45]. Although they avoid multiplying elements which are zero in one matrix, it is not possible to take advantage of sparsity in both matrices simultaneously [45]. In this work, we propose an algorithm, which will never perform a multiplication, if either value in the vectors being multiplied is zero. We also propose a slight modification to this algorithm, to improve speedup, in a lossy setting.

For speeding up image and video retrieval, several methods have been proposed for a bag-of-words model built on SIFT features [30, 35]. A Term Frequency-Inverse Document Frequency scheme was proposed to perform fast matching between visual words of the query image/video and database images/videos [35]. A vocabulary tree based quantization scheme which also performs indexing simultaneously was shown to improve retrieval results [30]. However, a Fisher Vector lies in a continuous high dimensional vector space. Therefore, approaches applicable for a bag-of-words model cannot be used to accelerate dot-products between Fisher-Vectors. To address this issue, a cascade architecture was proposed to accelerate dot-products between two vectors [3]. Since the first classifier needs to be applied on the complete dataset, it could only get a 2x improvement in run-time.

Web images have also been used for various tasks like concept discovery, event recognition, etc [9, 10]. Duan et al. [11] use weakly labeled web videos for event recognition by measuring the distance between two videos along with a transfer learning approach. Chen et al. [5] extract visual knowledge by using a semi-supervised approach. Sun et al. [37] use action names as the query and use downloaded web images for temporal localization of actions in videos. Further, to mitigate the domain shift between web data and the test set, many methods have been proposed using relevance feedback [21]. Jiang et al. [21] use easy samples first to perform re-ranking of the initial video list. For adapting detector weights, Tang et al. [38] gradually update the weights of web-trained detectors as they are applied to videos. Singh et al. [34] use the top ranked videos as positives, train a detector and use it to re-rank dataset videos. We also investigate relevance feedback, which can be efficiently implemented and, as our experiments will show, leads to significant improvements.



**Figure 1: Framework of VRFP.** The approach contains two parts: offline feature extraction (right) and online video retrieval (left). Compact representations of frame-level CNN features are built for database videos a priori. Given a new query, an on-line visual representation is constructed efficiently. Using a matching function, we compute the similarity between the query and database videos to obtain a ranked list. For domain adaptation, pseudo-relevance feedback is used to re-rank the initial list.

### 3. ON-THE-FLY RETRIEVAL APPROACH

The framework of VRFP is shown in Fig. 1. As a pre-processing step, a compact visual representation is constructed for every video in the dataset as shown in the right of Fig. 1. For any given text query, web search is used to collect images relevant to the query. An efficient visual representation for the query is then constructed based on those web images. The representation is constructed using CNN features extracted from video frames and web-images. Since both database videos and the text query are represented in the same vector space, we consider a number of techniques to compute their similarity. After computing similarity, a ranked list is constructed for the database videos. As the visual representation is built from web images, there will be a slight mismatch between features of the database set and web images - a domain shift. Thus, we construct a new visual representation for the query only on the basis of top ranked video features to re-rank the list. To this end, we will evaluate efficient ways of performing re-ranking on the test set.

#### 3.1 Compact Representations for Web Images and Videos

The process of building a visual representation for the query should be efficient, as our goal is real-time video retrieval. The final representation should not depend on the number of images returned by search engines or the number of concepts related to the query. Since a web-search returns a set of images and videos can be effectively represented as a set of frames [42], we generate a compact representation for both videos and web images by treating them both as unordered image sets. In this section, we discuss methods for building such compact representations.

**Average Pooling.** A common method to represent an image collection is average pooling, i.e., compute an average feature vector from the set of images. Formally, the feature vector of a video or an image collection  $X$  is  $f_{avg}(X) = \frac{1}{N} \sum_{i=1}^N \mathbf{x}_i$ , where  $N$  is the size of  $X$ , and  $\mathbf{x}_i$  is the feature vector of the  $i^{th}$  frame or image. Finally, this feature vector is normalized by its  $L_2$  norm.

**Fisher Vector (FV) Encoding.** Fisher Vector encoding first builds a  $K$ -component GMM model  $(\mu_i, \sigma_i, w_i : i = 1, 2, \dots, K)$  from training data, where  $\mu_i, \sigma_i, w_i$  are the mean, diagonal covariance, and mixture weights for the  $i^{th}$  component, respectively. Given a bag of features  $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T\}$ , its Fisher Vector is computed as:

$$\mathcal{G}_{\mu_i} = \frac{1}{T\sqrt{w_i}} \sum_{t=1}^T \gamma_t(i) \left( \frac{\mathbf{x}_t - \mu_i}{\sigma_i} \right) \quad (1)$$

$$\mathcal{G}_{\sigma_i} = \frac{1}{T\sqrt{2w_i}} \sum_{t=1}^T \gamma_t(i) \left( \frac{(\mathbf{x}_t - \mu_i)^2}{\sigma_i^2} - 1 \right) \quad (2)$$

where,  $\gamma_t(i)$  is the posterior probability. Then all  $\mathcal{G}_{\mu_i}$  and  $\mathcal{G}_{\sigma_i}$  are stacked to form the Fisher Vector. Following previous work [32], we compute a signed square-root on the Fisher Vector and then  $L_2$  normalization.

These pooling methods have been widely used to pool features from the same sample to form a vector representation for that sample, e.g., pooling frame-level features for one video, pooling SIFT features in an image, or pooling motion features like MBH in a video. It was recently shown that a Fisher Vector representation [42] on CNN features obtains state-of-the-art results for event classification. We propose to use the same representation for web-images, as they can be regarded as a stream of frames (like a video).

**Linear SVM.** A linear SVM [12] can be trained and used as a discriminative representation for web-images. They have been widely used in several on-the-fly retrieval techniques [2, 3] for image retrieval as they offer a good compromise between speed and accuracy.

#### 3.2 Matching Web Images and Videos

Given the representations of web images and database videos, we compute the similarity between them to perform video retrieval. Matching between a collection of web images and video frames can be performed in many ways. However, we need to ensure that the matching is efficient. If a generative model is not used to represent web images, a discriminative classifier like an SVM can be trained on image level features. Negative samples for a discriminative classifier are obtained by randomly sampling from background images. This classifier can then be applied on average pooled features of a video, and the response of this classifier can reflect their similarity. One disadvantage of using SVM for matching is that it considers each image as an individual training sample, instead of treating the search results as a collection of images (which seems closer to a video representation).

If we use the encoding schemes mentioned in the previous section, we can obtain a single vector representation for a collection

of web images and each database video. Thus, a simple way to measure similarity between the images and a video is to compute a cosine distance between their features; assuming features are  $L_2$  normalized:

$$S_i = \text{cosine}(\mathbf{f}_I, \mathbf{f}_{V_i}) \quad (3)$$

where  $\mathbf{f}_I, \mathbf{f}_{V_i}$  are the features of the image collection  $I$  returned by web search and the frames comprising the video  $V_i$ , respectively. Thus the database videos can be effectively ranked based on their similarity  $S_i$  with  $I$ <sup>1</sup>.

Although we use the Fisher Vector of CNN features as our representation, our matching process differs from previous work. In [42], a linear SVM is trained on Fisher Vectors of training videos. During on-the-fly retrieval, we only construct a single Fisher Vector from web-images. Therefore, we employ a nearest-neighbor approach for retrieval and will later show that this process can be accelerated without any loss in accuracy. This acceleration may not be possible if we use a linear SVM.

### 3.3 Re-ranking by Video Similarity

Since the visual representation for the query is constructed on web images, there is a domain shift between video and image representations. A common approach to deal with this type of domain shift is pseudo-relevance feedback. Pseudo relevance feedback commonly involves training an SVM on top ranked videos as positive samples and bottom ranked videos as negative samples [34]. Although other approaches have been proposed [21, 22], they take a few minutes to run at test time. Training an SVM may also take significant time, especially if it is trained on high dimensional features like Fisher Vectors. Moreover, most negative samples are not very informative due to the large diversity of background videos in the database.

We simply average the Fisher Vectors of the top ranked videos to obtain a representation of the query for pseudo-relevance. Cosine similarity is then computed with this mean vector to obtain the final ranked list. This is very efficient and robust to outliers due to averaging. Note that while re-ranking, it does not matter which representation was used to compute similarity between web images and video frames. This is because we build a representation for pseudo relevance based on top ranked videos and matching can now be performed between representations which are constructed using similar features.

## 4. FAST FISHER VECTOR PRODUCTS

Computing the dot-product between two Fisher Vectors can be slow due to their high dimensionality. Hence, in this section, we describe a Fast Fisher Vector Product (FFP) algorithm which takes advantages of the sparsity of Fisher Vectors and accelerates the dot-product procedure without loss in accuracy.

### 4.1 Sparsity in FVs of Videos and Web-Images

FVs constructed from SIFT features extracted on images are dense (around 50% of the entries are non-zero) [33] due to significant variation in appearance of small patches within an image. However, due to temporal continuity between frames, CNN features of video frames are quite similar. Moreover, search results returned by web image search can contain very similar subsets. Hence, in both these cases, it is likely that many images contribute to the same set

<sup>1</sup>We investigated other similarity metrics, e.g., histogram intersection, and Canonical Correlation Analysis. However, cosine distance gives the best performance with limited computational cost.

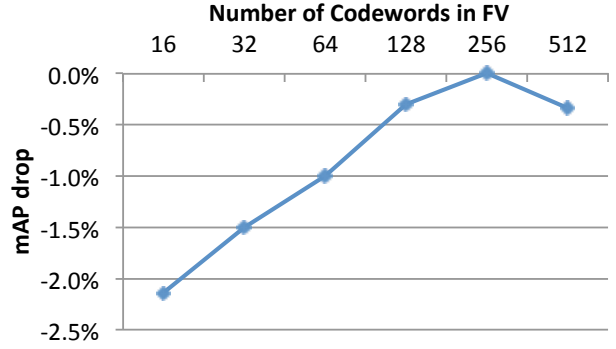


Figure 2: Drop in mAP for the TRECVID MED13 data for different codeword sizes

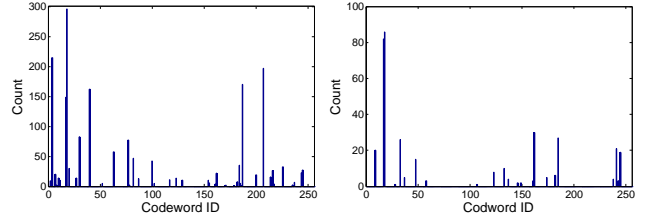


Figure 3: Histogram showing how many frames or images contribute to each codeword in the FV. The plot on the left is constructed using web-images returned for a query and the one on the right is for a video in the TRECVID MED13 dataset.

of codewords. Therefore, we expect the final Fisher Vector to be sparse (we find that only 10% of the entries are non-zero when  $K$  is large). In Fig. 3 we present a plot showing the histogram of how many frames or images contribute to each codeword ( $K = 256$ ) in two Fisher Vectors (one constructed on a web-image collection and the other on frames of one video). Although in theory each sample (video frame/web-image) contributes to every codeword in a Fisher Vector, the probabilities for most codewords are so low that they fall below measurable precision. Even if only a single sample contributes to a codeword, we call it a *non-zero codeword* in a Fisher Vector. Also note that high dimensional Fisher Vectors with more codewords improve performance as in Fig. 2. So, even though the representation is sparse, it performs better than a low dimensional representation using fewer codewords.

Therefore, a simple method to accelerate the computation of dot product between Fisher Vectors would be to use a sparse matrix representation, which only stores the non-zero elements in the matrix and their indices. Even if we multiply a sparse matrix with a column vector, theoretically, we expect to observe a linear speedup with respect to the number of zeros in the matrix. However, this operation only takes advantage of sparsity in the matrix or the vector. It would still need to multiply some non-zero elements with zeros in the vector or vice-versa [45].

Using a sparse representation would also reduce the memory required for storing the FVs. For example, if 90% of the codewords of the FV are zero, it would lead to reduction in memory required by a factor of 10. For a general matrix, we would need to store all indices of the non-zero entries, but this is not required for FVs. This is because zeros are not randomly distributed in a FV, but instead have a grouping structure. If no video frame or web-image  $x_i$  contributes to a codeword  $i$  (i.e. the codeword has an extremely low probability which falls below measurable precision for all samples), then all values for that codeword  $\mathcal{G}_{\mu_i}, \mathcal{G}_{\sigma_i}$  are zero. This trick



was proposed in [33] for compressing sparse FVs. We will show that this property can also be used to accelerate the computation of dot-products between sparse FVs.

## 4.2 Lossless Matching Algorithm

Let  $Q, V_i \in \mathbf{R}^N$  denote the FV corresponding to web-images and video frames respectively. Let  $T$  denote the database of videos such that,  $T = \{V_1, V_2, \dots, V_M\}$ , where  $M$  is the number of videos. Assume the FV is constructed from a GMM of  $K$  codewords, where each codeword is of dimension  $D$ . Since the FV includes differences from the mean and the variance for each codeword,  $N = 2DK$ . Compute  $I^T = \{I_1^T, I_2^T, \dots, I_j^T, \dots, I_M^T\}$ , where,  $I_j^T \subseteq \{1, 2, \dots, K\}$  denotes the set of non-zero codeword indices in  $V_j$ . Let  $I^Q$  denote the set of non-zero codewords indices in the Fisher Vector for query  $Q$ . Once the query is provided, compute  $I^Q$  by checking if the sub-vector corresponding to each codeword is non-zero. Compute the set intersection  $S^T = \{S_1^T, S_2^T, \dots, S_j^T, \dots, S_M^T\}$ , where,  $S_j^T = I_j^T \cap I^Q$ . Finally, compute the dot product  $P_j$  between  $Q$  and  $V_j$ , only for codewords which are in  $S_j^T$ .

## 4.3 Analysis

First, assume that each codeword across the video database is equally probable to be a *non-zero codeword* with a probability  $p^T$ . We also assume that each codeword in the event query is equally probable to be a *non-zero codeword* with a probability  $p^Q$ . Therefore, the probability that a codeword contributes a non-zero value to the dot-product is  $p^T p^Q$ . Thus, the expected number of operations for performing a dot-product between a video and a query Fisher Vector is  $2KDp^T p^Q$ . The intersection only needs to be computed between codeword indices ( $I^T$  and  $I^Q$ ). So, after we obtain  $I_j^T$  and  $I^Q$ , it only takes  $\min(|I_j^T|, |I^Q|)$  operations to construct  $S_j^T$  (using a hash-table for the codeword indices). Thus, the expected number of operations for calculating  $S_j^T$  is  $K \min(p^T, p^Q)$ . Therefore, the expected number of operations required for matching in VRFP is  $K \min(p^T, p^Q) + 2KDp^T p^Q$ . The expected speedup over the brute force dot product algorithm is given by:

$$ES_{unbiased} = \frac{2KD}{2KDp^T p^Q + K \min(p^T, p^Q)} \quad (4)$$

$$= \frac{1}{p^T p^Q + \frac{\min(p^T, p^Q)}{2D}}$$

If  $C_1$  elements of  $T$  are non-zero and  $C_2$  elements of  $Q$  are non-zero, then  $p^T$  and  $p^Q$  can be approximated as,  $p^T = \frac{C_1}{MN}$  and  $p^Q = \frac{C_2}{N}$  respectively. This shows that the number of arithmetic operations required by the algorithm is quadratic in terms of sparsity in Fisher Vectors (assuming intersection computation time is bounded by a constant due to large  $D$ ). Note that sparsity is defined as the proportion of non-zeros in a matrix. In the case of sparse matrix multiplication, intersection computation is not efficient, as in the general case the grouping structure present in Fisher Vectors is absent. Therefore, intersection computation would take  $2KD \min(p^T, p^Q)$  operations. Consequently, the number of arithmetic operations required for performing sparse multiplication would be linear in terms of sparsity in Fisher Vectors. The time complexity for different algorithms is shown in Table 1.

This analysis is valid when all codewords are equally probable to be non-zero. However, there can be a few codewords which are non-zero most of the time. In this case, the previous analysis will not hold. Suppose  $K - X$  ( $X \ll K$ ) codewords are equally probable to be non-zero with a low probability  $p_l$ , while  $X$  of them are equally probable to be non-zero with a high probability

$p_h$ . For simplicity, assume that the sparsity patterns of web-images and video frames are similar, i.e.,  $p_l^T = p_l^Q$  and  $p_h^T = p_h^Q$ . The expected speedup per codeword (ignoring the intersection computation time) in this case would be:

$$ES_{biased} = \frac{K}{Xp_h^2 + (K - X)p_l^2} \quad (5)$$

**Table 1: Matrix Multiplication (MM) vs Fast FV Products**

Method	Naive MM	Sparse MM	FFP
Complexity	$\mathcal{O}(N)$	$\mathcal{O}(\min(p^Q, p^T)N)$	$\mathcal{O}(p^Q p^T N)$

## 4.4 Lossy Matching Algorithm

Generally,  $p_l$  is around 0.1 (10% of the elements are non-zero), therefore  $p_l^2$  is very small. However, even if  $p_h$  is 0.5, it would significantly slow down the algorithm. Therefore, a simple trick to speedup the algorithm would be to remove the codewords which are non-zero with a very high probability. As we will show in our experiments, with 256 codewords, removing only 2 codewords can yield a speedup of 20% over the matching algorithm without a significant loss in accuracy. In the case of normal multiplication, removing 2 codewords would have given a speedup of less than 1%.

## 5. EXPERIMENTAL RESULTS

### 5.1 Datasets

We evaluate our method on three challenging event detection datasets.

**TRECVID MED13/14 EK0.** The TRECVID MED 2013/2014 EK0 dataset consist of unconstrained Internet videos collected by the Linguistic Data Consortium from various Internet video web sites. Each video contains only one complex event or content not related to any event. There are in total 30 complex events in this dataset: “E1: birthday party”, “E2: changing a vehicle tire”, “E3: flash mob gathering”, “E4: getting a vehicle unstuck”, “E5: grooming an animal”, “E6: making a sandwich”, “E7: parade”, “E8: park-our”, “E9: repairing an appliance”, “E10: working on a sewing project”, “E11: attempting a bike trick”, “E12: cleaning an appliance”, “E13: dog show”, “E14: giving directions to a location”, “E15: marriage proposal”, “E16: renovating a home”, “E17: rock climbing”, “E18: town hall meeting”, “E19: winning a race without a vehicle”, “E20: working on a metal crafts project”, “E21: bee keeping”, “E22: wedding shower”, “E23: non-motorized vehicle repair”, “E24: fixing musical instrument”, “E25: horse riding”, “E26: felling a tree”, “E27: parking a vehicle”, “E28: playing fetch”, “E29: tailgating”, “E30 tuning musical instrument”. Videos of the first 20 events together with background videos (around 23,000 videos), form a test set of 25,000 videos for MED13 testset, and last 20 events with similar background videos form MED14 testset. In the EK0 setting, no ground-truth training videos are available. We apply on the dataset videos, and mAP (mean Average Precision) score is calculated based on the video ranking.

**Columbia Consumer Videos (CCV)[25].** The CCV dataset contains 9,317 videos collected from YouTube with annotations of 20 semantic categories: “E1: basketball”, “E2: baseball”, “E3: soccer”, “E4: ice skating”, “E5: skiing”, “E6: swimming”, “E7: biking”, “E8: cat”, “E9: dog”, “E10: bird”, “E11: graduation”, “E12: birthday”, “E13: wedding reception”, “E14: wedding ceremony”, “E15: wedding dance”, “E16: music performance”, “E17: non-music performance”, “E18: parade”, “E19: beach”, “E20: playground”. This dataset is evenly split into 4,659 training videos and

4,658 dataset videos. Since we focus on the scenario where no training videos are available, we only run our method on the dataset videos and calculate the mAP.

## 5.2 Implementation Details

For each category in the dataset, we use the name of the query to search for and download images from Google. After obtaining the web image collection, we build our web image representation. On average, around 700 images are downloaded for each event. We sample one frame every 2 seconds for TRECVID MED and CCV dataset. The implementation of AlexNet [27] provided in Caffe [20] is utilized to extract 4,096 dimensional fc7 layer features for web images and video frames.

For building Fisher Vectors, we first reduce the original features to 256 dimension using principal component analysis (PCA), and then use 256 components for Fisher Vectors. We learn the PCA projection and Fisher Vector GMM components on TRECVID background videos. Finally, VLFeat [39] is used to generate Fisher Vectors.

We also apply another square root normalization on the Fisher Vector built from video frames (when comparing similarity between web images and video frames). This is because a video level Fisher Vector is more unevenly distributed when compared to Fisher Vectors built on web images. To boost sparsity, we shrink values whose magnitude falls below  $10^{-3}$  in the FV to zero. We use the top 50 ranked videos in the initial ranked list for re-ranking. When SVM is used for re-ranking, the bottom 1,000 videos are used as negatives. Liblinear [12] is used to train and test all these linear SVMs, and the  $C$  parameter is set to 1 for all SVMs.

In total, it takes less than a second for VRFP to perform retrieval for 100,000 videos: downloading images from the web takes less than 500 milliseconds, CNN features on 700 images are extracted in 150 milliseconds (using AlexNet implementation of Torch on Titan X), building Fisher Vector of web images takes 30ms, and feature matching and re-ranking is performed in less than 240 milliseconds when using 130,072 dimensional Fisher Vectors. A compressed Fisher Vector representation (using sparsity) occupies only 1GB memory for 100,000 videos (with 16 bit floating point precision). All performance experiments were carried out on an Intel Ivy Bridge E5-2680v2 processor using only a single core.

For fairness when comparing running time of all methods, double precision was used for non-integer storage (as MATLAB sparse matrix multiplication requires a double input). Other than sparse matrix multiplication, every baseline like Product Quantization, High Variance Subspaces First, naive matrix multiplication and the proposed fast FV product code was written in C++. Each code was optimized taking cache locality into account. Since our method may produce different intersections for each query, the computation time can vary per query. Therefore, we average the time required for all 20 queries in TRECVID MED13 in our results.

## 5.3 Performance of Different Representations for Videos and Images

We compare different image and video representations in Table 2. For a fair comparison, these results are before the re-ranking step. Wherever SVM is not mentioned (Table 2), we use cosine similarity to measure distance between the representations of web images and dataset videos. Otherwise, an SVM was trained using the web images as positives and 1,000 randomly selected web images of other queries as negatives. Note that training an SVM on a large number of samples can take a significant amount of time. It takes 300ms to train a linear SVM, while building FV only needs 30ms. However, we provide this comparison for completeness.

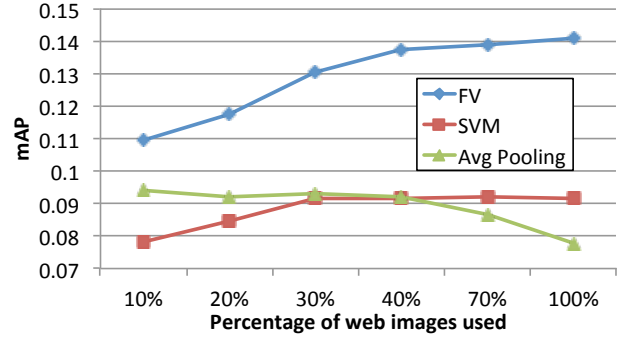


Figure 4: TRECVID MED13 mAP of FV, Average Pooling, and linear SVM when gradually adding more web images

Table 2 shows that SVM and Fisher Vector work better than average pooling in all cases. Since TRECVID MED13/14 are complex event detection tasks and contain diverse videos, using a Fisher Vector representation for both web images and videos yields the best performance for matching. However, on the CCV dataset, SVM using average pooled features performs better. This is because the CCV dataset contains videos with only a few very similar frames. Since CCV contains very simple events, the retrieved web images also have similar appearance. Nevertheless, in section 5.5, we will show that the video retrieval performance of Fisher Vector outperforms SVM on the CCV dataset after re-ranking.

Table 2: Comparison among different representations

Method	MED13	MED14	CCV
Avg Pooling	7.78%	2.68%	28.81%
Avg + SVM	9.17%	6.0%	<b>37.28%</b>
Fisher Vectors	<b>14.1%</b>	<b>8.49%</b>	32.65%

## 5.4 Robustness Analysis of Representations

In this section, we conducted experiments on the TRECVID MED13 dataset to show the advantages of the Fisher Vector representation compared to average pooling and SVM for matching noisy web image collection with video frames. We investigate the following key factors that may affect retrieval accuracy and evaluate the robustness of different representations:

**Number of Web Images** In Fig. 4, we vary the number of web images for building the representations and training SVM, i.e., only the top 10%, 20%, 30%, 40%, 70%, and 100% retrieved images from the search engine are used. We show the mAP on MED13 test dataset for each case.

From this figure, we see that the performance of both FV and SVM improves significantly until top 30% web images are used. When more web images are added, the mAP of SVM does not improve and even drops at the end of the plot. This is because the top ranked results returned by an image search engine contain fewer noisy images, while the low ranked images are often more noisy and sometimes completely irrelevant to the query. Thus, using these noisy images as positive samples to train the SVM hurts performance. This problem is more severe for average pooling - its mAP drops consistently when more images are used to calculate the average of features. This is because the top 10% of the web images share similar appearance, and average pooling of these images serves as a nearest mean classifier to these images. However, when more images are used, diverse results (including outliers) affect the mean estimation and hurt performance.

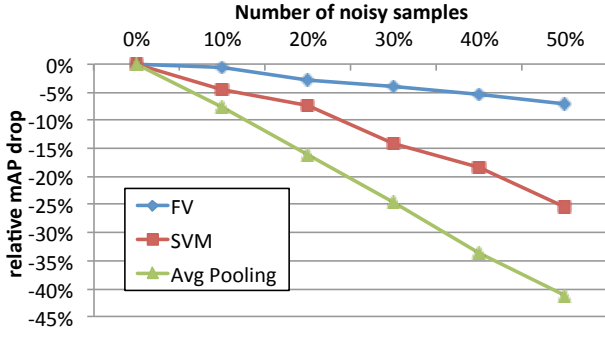


Figure 5: Relative mAP drop of FV, SVM and average pooling when gradually adding negative images from other queries.

Table 3: MAP improvement for MED13 dataset by outlier removal

Method	Avg Pooling	Avg + SVM	FV
with outliers	7.78%	9.17%	14.10%
without outliers	8.60%	9.92%	14.25%
Relative change	10.54%	8.18%	1.06%

In contrast to SVM and average pooling, the FV representation always benefits from more training images even when some of them are noisy. This means that FV is a better choice when calculating similarity between a web image collection and frames of a video.

**Outliers in Web Images** Outliers in image search results is a well-known issue for models trained on web images. Some works [2, 4] show that by identifying and eliminating outliers, the web-supervised classifier yields higher accuracy for image/video retrieval. In this paper, we employ state-of-the-art outlier removal algorithms based on an autoencoder [41]. For web images of a query whose feature vectors are  $\{x_1, x_2, \dots, x_n\}$ , an autoencoder minimizes the sum of reconstruction error:

$$\mathcal{J}(f) = \sum_i \epsilon_i = \sum_i \|f(x_i) - x_i\|^2 \quad (6)$$

where  $f(\cdot)$  is a neural network with a single hidden layer, and  $\epsilon_i$  denotes the squared loss of sample  $x_i$ . The main idea is that if we train an autoencoder using the web images for a query, the positive samples have a smaller reconstruction error than outliers. Thus, if we apply 2-means clustering on image reconstruction errors, the outliers would be in the cluster with a higher reconstruction error. The outlier removal algorithm improves the mAP for all representations as shown in Fig. 3. For FV, the outlier removal only gives a marginal improvement, which means the FV is more robust to the outliers in web image search. Thus, FV is more suitable for building representations for noisy data like web images, and an outlier removal algorithm is not necessary to achieve high accuracy. Therefore, we do not include this outlier removal method in our final results. This also avoids the need to train an autoencoder after the web images are obtained, which hurts real-time performance.

**Noisy Samples** To further illustrate the superiority of FV representation, we evaluate FV and the other two representations by adding 10% to 50% negative images to the positive training data. More specifically, for SVM, we sample some images in the negative training set and change their labels to positive and re-train the SVM; for FV and average pooling, these negatives, together with the original positives, are used to build the feature vectors and av-

Table 7: Comparison among different representations

Method	MED13	MED14	CCV
FV (No re-ranking)	14.10%	8.49%	32.65%
Avg + SVM	11.95%	6.28%	36.29%
FV + SVM	15.91%	8.72%	38.08%
FV similarity	<b>16.44%</b>	<b>9.67%</b>	<b>40.81%</b>

eraging. The results are shown in Fig. 5. We can see that when more and more negative images are used, the performance drop of FV is less than SVM, while average pooling suffers the most from negative samples.

Consequently, we conclude that FV is a better choice than SVM and average pooling for on-the-fly video retrieval using web images.

## 5.5 Exploring Re-ranking Methods

We show the effectiveness of re-ranking and compare three different re-ranking strategies in Table 7. Re-ranking helps in all three datasets. The most efficient and best performing method for this task is computing an average Fisher Vector for top ranked videos and then computing its cosine similarity with the dataset videos (FV Similarity). Training a linear SVM classifier on Fisher Vectors using top ranked videos as positives and bottom ranked videos as negatives (FV + SVM) and re-ranking using the classifier response can achieve similar performance as FV similarity. However, it takes more than 2 seconds to train an SVM due to high dimensional features. Some other re-ranking methods like [22, 21] take around 2.5 minutes to run. Thus, using methods like SVM for re-ranking are not suitable for fast and accurate video retrieval. Apart from computing cosine similarity between Fisher Vectors, we also consider training an SVM on average pooled CNN features (Avg + SVM). However, this does not work as well as Fisher Vectors since features of different videos can be very different and averaging the features further leads to a loss of information.

For the CCV dataset, re-ranking the initial results generated by FV cosine similarity improves results mAP to 40.81%. However, if we use the same re-ranking for Avg + SVM method, the mAP only improves from 37.28% to 38.73%, which is lower than the re-ranking result of FV (re-ranking improve FV from 32.65% to 40.81%). The videos in the initial top ranked list obtained using FV are diverse in appearance, so the average FV of top-ranked videos contains more useful information. Thus, FV is generally a good choice to match web images and video frames and to re-rank the dataset videos.

## 5.6 Speedup using Fast Fisher Vector Products

As shown in Fig. 3, the Fisher Vectors of web-images and video frames are sparse. For web-images we find that only 15% of the code words are non-zero in all 20 queries for TRECVID MED13. For video Fisher Vectors, only 7% of the video frames are non-zero. Therefore, just using sparse matrix multiplication, we obtain a speedup of more than 11 (360ms vs 4000ms) over naive matrix multiplication (NMM). When we use the proposed Fast Fisher Vector products, we obtain an additional 6x speedup over sparse matrix multiplication (60ms vs 360ms, 66x NMM). This is because as we increase sparsity linearly, our computation reduces quadratically. For sparse matrix multiplication, computation only decreases linearly with sparsity. Surprisingly, even our lossless version of Fast Fisher Vector Products is 2x faster than computing a naive dot product between 4096 dimension CNN features. Using the lossy

**Table 4: AP for each event in TRECVID MED13 EK0 dataset**

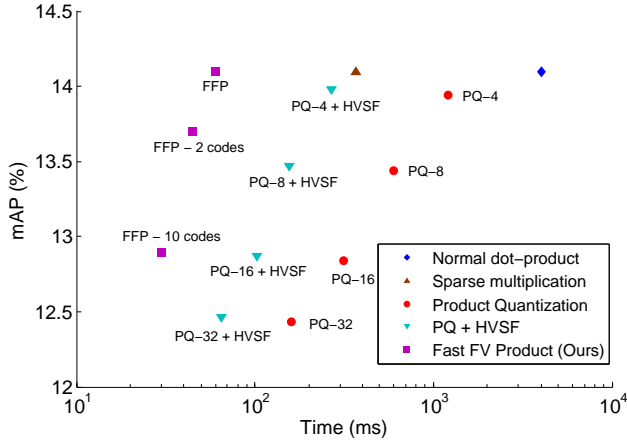
MED13 EK0	E1	E2	E3	E4	E5	E6	E7	E8	E9	E10	E11	E12	E13	E14	E15	E16	E17	E18	E19	E20	mAP
EventNet [43]	9.7	32	0.3	1	1.8	5.7	27.4	18.1	4.3	0.9	0.8	2.9	<b>47.0</b>	0.1	0.5	0.3	<b>7.5</b>	<b>16.1</b>	0.1	0.5	8.86
Pair-Concept [34]	<b>17.2</b>	43.8	42.2	<b>50.3</b>	6.43	12.6	15.8	6.56	13.7	3.36	8.14	<b>7.94</b>	1.11	0.36	0.26	<b>3.13</b>	1.73	1.32	0.13	1.02	11.8
Concept Proto [28]	15.4	32	27.1	40.6	9.5	16.4	24	11.2	21.3	8.9	6.1	2.6	1.1	0.8	0.5	2.6	3.6	3.5	<b>10.1</b>	<b>1.4</b>	11.9
TagBook [29]	15.5	33.7	17.4	31.2	<b>20.1</b>	9.9	18.5	<b>21.5</b>	21.1	<b>9.8</b>	6.6	2.3	20.0	0.5	0.3	1.8	2.6	14.8	9.9	0.2	12.9
VRFP web-only (Ours)	9.4	42.7	38.5	42.4	7.6	13.7	26.9	15.5	17.0	3.7	9.2	1.6	33.7	<b>1.4</b>	0.1	2.7	4.9	8.4	0.8	1.1	14.1
VRFP re-ranking(Ours)	10.8	<b>44.4</b>	<b>42.7</b>	50.2	7.5	<b>19.3</b>	<b>30.1</b>	11.4	<b>31.5</b>	2.2	<b>17.2</b>	1.9	45.3	0.6	<b>0.9</b>	1.4	7.4	1.5	1.6	0.6	<b>16.4</b>

**Table 5: AP for each event in TRECVID MED14 EK0 dataset**

MED14 EK0	E11	E12	E13	E14	E15	E16	E17	E18	E19	E20	E21	E22	E23	E24	E25	E26	E27	E28	E29	E30	mAP
TagBook [29]	7.5	8.0	15.7	0.6	0.5	<b>4.7</b>	2.0	<b>12.0</b>	6.3	0.5	0.9	<b>3.5</b>	<b>26.5</b>	0.9	11.8	<b>7.2</b>	3.5	<b>3.5</b>	0.6	0.9	5.9
AutoVisual [24]	5.4	<b>15.0</b>	<b>42.4</b>	<b>2.5</b>	<b>4.1</b>	0.5	<b>11.9</b>	1.9	<b>6.6</b>	<b>6.4</b>	45.65	3.0	1.3	1.8	<b>13.24</b>	0.5	4.0	0.2	0.1	1.1	8.4
VRFP web-only (Ours)	9.4	2.6	29.2	0.7	1.1	2.7	5.0	9.4	1.1	1.1	<b>70.2</b>	1.0	3.6	6.3	6.6	1.9	7.9	2.4	<b>2.0</b>	5.8	8.5
VRFP re-ranking (Ours)	<b>17.6</b>	3.1	39	0.4	1.1	1.5	7.4	3.5	2.3	0.5	65.47	0.4	4.9	<b>15.7</b>	7.4	2.0	<b>12.0</b>	1.3	1.5	<b>6.4</b>	<b>9.7</b>

**Table 6: AP for each event in CCV dataset**

CCV	E1	E2	E3	E4	E5	E6	E7	E8	E9	E10	E11	E12	E13	E14	E15	E16	E17	E18	E19	E20	mAP
Large Concept [6]	5.8	7.9	12.5	7.4	25.4	21.5	11.3	8.8	18.7	15.6	24.2	10.7	<b>29.3</b>	22.1	22.7	21.4	16	21.9	47	10.1	18
EventNet [43]	<b>59.5</b>	<b>55</b>	<b>57.5</b>	<b>77.7</b>	69.1	73.6	19.6	41.9	29	10.3	15.1	<b>13.1</b>	12.9	<b>27</b>	18.4	28.7	<b>35.6</b>	<b>47.4</b>	16.4	3.89	35.6
VRFP web-only (Ours)	11.0	38.1	34.8	41.8	57.9	73.4	49.0	37.7	27.2	<b>22.5</b>	32.0	9.82	7.02	9.84	36.0	<b>31.0</b>	12.0	37.4	55.6	28.9	32.7
VRFP re-ranking (Ours)	34.3	53.4	54.7	50.2	<b>75.4</b>	<b>80.1</b>	<b>65.2</b>	<b>49.0</b>	<b>42.3</b>	19.3	<b>32.1</b>	9.69	8.26	8.66	<b>49.42</b>	24.4	15.0	31.2	<b>69.3</b>	<b>44.1</b>	<b>40.8</b>



**Figure 6: mAP vs. matching time on TRECVID MED13 dataset.  $PQ-k$  means Product Quantization with subvector length equal to  $k$ . HVSF stands for the High Variance Space First algorithm described in [44].  $FFP - k$  codes is our fast FV product method with  $k$  codewords removed.**

algorithm, we obtain a speedup of 2x (30ms, 132x NMM) over the lossless algorithm just by removing 10 codewords (< 4% of whole codeword size). This is because high probability codewords dominate the sum in the denominator of Equation 5.

We also compare our algorithm with other state-of-the-art algorithms like product quantization (PQ) [17] and High Variance Subspaces First (HVSF) [44] which are used for fast retrieval of high dimensional vectors. We show results for product quantization and HVSF with 4 different sub-vector lengths. In HVSF, we select the top 20% subspaces with higher variances to calculate the initial ranked list and compute the extract dot-product for top 500 videos [44]. Accuracy of PQ and HVSF drops when we increase the sub-

vector length, although they do provide linear speedup. Fig. 6 compares the accuracy and efficiency of our system without re-ranking with these methods. Note that PQ and HVSF are approximate algorithms, which reduce accuracy. Our lossless algorithm can obtain a 66x speedup over standard matrix multiplication without any loss in accuracy. If the dot product in the remaining codewords in  $S^T$  needs to be accelerated, PQ can be applied on top of VRFP. To further reduce the elements in  $S^T$ , intersection can be computed with the subspaces in HVSF. The aim of the proposed algorithm is not be an alternative to PQ or HVSF, but to remove redundant multiplications which happen when computing dot products between sparse Fisher Vectors.

**Table 8: Comparative results on TRECVID MED13 EK0 and CCV**

Method	MED13 EK0	CCV
Large Concept Bank [6]	2.2%	18.0%
Concept Discovery [4]	2.3%	-
Composite Concept [15]	6.4%	-
AutoVisual [24]	7.4%	-
EventNet [43]	8.86%	35.6%
MMPRF [22]	10.1%	-
Pair-Concept [34]	11.8%	-
Concept Prototypes [28]	11.9%	-
Multi-Modal [40]	12.6%	-
TagBook [29]	12.9%	-
SPaR [21]	12.9%	-
<b>VRFP(Ours)</b>	<b>16.4%</b>	<b>40.8%</b>

## 5.7 Comparison with Other Methods

Table 8 and 9 show the results of VRFP on TRECVID MED EK0 dataset and CCV dataset. Compared with other automatic methods, our method performs favourably. In contrast to all these methods, VRFP only requires a short query, instead of a long detailed description. Since the semantic gap is large for pre-defined con-



**Table 9: Comparative results on TRECVID MED14 EK0 dataset**

Method	MED14 EK0
TagBook [29]	5.9%
AutoVisual [24]	8.37%
<b>VRFP(Ours)</b>	<b>9.67%</b>

cept based approaches like [40, 28, 29], their performance is lower when compared to VRFP. Although methods like [4, 34] discover concepts after the query is given, VRFP compares favourably because we use a better representation for matching web images and video frames. Note that our pipeline is fully automatic and does not require any manual intervention. Methods like [40, 22] also use multi-modal features like automatic speech recognition (ASR), OCR or motion features like improved trajectories, etc. Note that for [24], we only report its automatic version instead of the one with manual inspection for fair comparison. Further, as shown in the previous section, our runtime performance is at least 10,000 times faster than the methods which leverage web data after the query is given, like [4, 34].

In Table 4, 5 and 6 we also show the AP scores of all events in TRECVID MED13/14 and CCV dataset, where VRFP with and without re-ranking are shown. These tables show that our method performs better and re-ranking helps in improving results.

## 6. CONCLUSION

We proposed VRFP - an on-the-fly video retrieval system with only a short text input query. Without any pre-defined concepts, VRFP retrieves web images for that query and builds a bag-based scalable representation via a Fisher Vector. Simple matching and re-ranking strategies that do not require discriminative training are used to perform video retrieval. We showed that a Fisher Vector is robust to noise present in web-images and also studied how it performs as we increase the number of web-images for training. To accelerate the computation of dot-product between high dimensional Fisher Vectors, we presented a lossless algorithm in which the number of arithmetic operations required are quadratic in terms of sparsity in Fisher Vectors. State-of-the-art results on three popular event retrieval datasets demonstrated the effectiveness of our approach.

## Acknowledgement

This work is supported by the Intelligence Advanced Research Projects Activity (IARPA) via the Department of Interior National Business Center contract number D11PC20071. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes not with standing any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/NBC, or the U.S. Government. The authors would like to thank Yin Cui, Guangnan Ye, and Yitong Li for providing the AP for each event of their methods. The authors acknowledge the University of Maryland supercomputing resources <http://www.it.umd.edu/hpcc> made available for conducting the research reported in this paper.

## 7. REFERENCES

- [1] R. Arandjelović and A. Zisserman. Multiple queries for large scale specific object retrieval. In *British Machine Vision Conference*, 2012.
- [2] K. Chatfield, R. Arandjelović, O. M. Parkhi, and A. Zisserman. On-the-fly learning for visual search of large-scale image and video datasets. *International Journal of Multimedia Information Retrieval*, 2015.
- [3] K. Chatfield and A. Zisserman. Visor: Towards on-the-fly large-scale object category retrieval. In *Asian Conference on Computer Vision*, Lecture Notes in Computer Science. Springer, 2012.
- [4] J. Chen, Y. Cui, G. Ye, D. Liu, and S.-F. Chang. Event-driven semantic concept discovery by exploiting weakly tagged internet images. In *Proceedings of International Conference on Multimedia Retrieval*, page 1. ACM, 2014.
- [5] X. Chen, A. Shrivastava, and A. Gupta. Neil: Extracting visual knowledge from web data. In *IEEE International Conference on Computer Vision (ICCV)*, 2013.
- [6] Y. Cui, D. Liu, J. Chen, and S.-F. Chang. Building a large concept bank for representing events in video. *arXiv preprint arXiv:1403.7591*, 2014.
- [7] J. Dalton, J. Allan, and P. Mirajkar. Zero-shot video retrieval using content and concepts. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*. ACM, 2013.
- [8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.
- [9] S. K. Divvala, A. Farhadi, and C. Guestrin. Learning everything about anything: Webly-supervised visual concept learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.
- [10] L. Duan, D. Xu, and S.-F. Chang. Exploiting web images for event recognition in consumer videos: A multiple source domain adaptation approach. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.
- [11] L. Duan, D. Xu, I.-H. Tsang, and J. Luo. Visual event recognition in videos by learning from web data. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(9):1667–1680, 2012.
- [12] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*.
- [13] R. Fergus, L. Fei-Fei, P. Perona, and A. Zisserman. Learning object categories from google’s image search. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 1816–1823. IEEE, 2005.
- [14] Y. Gong and S. Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *Computer Vision and Pattern Recognition (CVPR)*, 2011 *IEEE Conference on*, pages 817–824. IEEE, 2011.
- [15] A. Habibian, T. Mensink, and C. G. Snoek. Composite concept discovery for zero-shot video event detection. In *Proceedings of International Conference on Multimedia Retrieval*, page 17. ACM, 2014.
- [16] A. Habibian, T. Mensink, and C. G. Snoek. Videostory: A new multimedia embedding for few-example recognition and translation of events. In *Proceedings of the ACM International Conference on Multimedia*. ACM, 2014.
- [17] H. Jegou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(1):117–128, 2011.
- [18] H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating

- local descriptors into a compact image representation. In *Computer Vision and Pattern Recognition (CVPR)*, 2010 *IEEE Conference on*, pages 3304–3311. IEEE, 2010.
- [19] H. Jégou, F. Perronnin, M. Douze, J. Sanchez, P. Perez, and C. Schmid. Aggregating local image descriptors into compact codes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(9):1704–1716, 2012.
- [20] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.
- [21] L. Jiang, D. Meng, T. Mitamura, and A. G. Hauptmann. Easy samples first: Self-paced reranking for zero-example multimedia search. In *ACM MM*, 2014.
- [22] L. Jiang, T. Mitamura, S.-I. Yu, and A. G. Hauptmann. Zero-example event search using multimodal pseudo relevance feedback. In *Proceedings of International Conference on Multimedia Retrieval*. ACM, 2014.
- [23] L. Jiang, S.-I. Yu, D. Meng, T. Mitamura, and A. G. Hauptmann. Bridging the ultimate semantic gap: A semantic search engine for internet videos. In *International Conference on Multimedia Retrieval*, 2015.
- [24] L. Jiang, S.-I. Yu, D. Meng, Y. Yang, T. Mitamura, and A. G. Hauptmann. Fast and accurate content-based semantic search in 100m internet videos. In *Proceedings of the 23rd Annual ACM Conference on Multimedia Conference*, pages 49–58. ACM, 2015.
- [25] Y.-G. Jiang, G. Ye, S.-F. Chang, D. Ellis, and A. C. Loui. Consumer video understanding: A benchmark database and an evaluation of human and machine performance. In *Proceedings of the 1st ACM International Conference on Multimedia Retrieval*, page 29. ACM, 2011.
- [26] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *Computer Vision and Pattern Recognition (CVPR)*, 2014 *IEEE Conference on*, pages 1725–1732. IEEE, 2014.
- [27] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- [28] M. Mazloom, A. Habibian, D. Liu, C. G. Snoek, and S.-F. Chang. Encoding concept prototypes for video event detection and summarization. 2015.
- [29] M. Mazloom, X. Li, and C. G. Snoek. Tagbook: A semantic video representation without supervision for event detection. *arXiv preprint arXiv:1510.02899*, 2015.
- [30] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, volume 2, pages 2161–2168. IEEE, 2006.
- [31] F. Perronnin and C. Dance. Fisher kernels on visual vocabularies for image categorization. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.
- [32] F. Perronnin, J. Sánchez, and T. Mensink. Improving the Fisher Kernel for Large-Scale Image Classification. In *., editor, ECCV*, volume 6314 of *., pages 143–156. ., 2010.*
- [33] J. Sánchez and F. Perronnin. High-dimensional signature compression for large-scale image classification. In *Computer Vision and Pattern Recognition (CVPR)*, 2011 *IEEE Conference on*, pages 1665–1672. IEEE, 2011.
- [34] B. Singh, X. Han, Z. Wu, V. Morariu, and L. Davis. Selecting relevant web trained concepts for automated event retrieval. In *IEEE International Conference on Computer Vision (ICCV)*, 2015.
- [35] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1470–1477. IEEE, 2003.
- [36] K. Soomro, A. R. Zamir, and M. Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.
- [37] C. Sun, S. Shetty, R. Sukthankar, and R. Nevatia. Temporal localization of fine-grained actions in videos by domain transfer from web images. *ACM MM*, 2015.
- [38] K. Tang, V. Ramanathan, L. Fei-Fei, and D. Koller. Shifting weights: Adapting object detectors from image to video. In *Advances in Neural Information Processing Systems*, 2012.
- [39] A. Vedaldi and B. Fulkerson. Vlfeat: An open and portable library of computer vision algorithms. In *Proceedings of the international conference on Multimedia*, pages 1469–1472. ACM, 2010.
- [40] S. Wu, S. Bondugula, F. Luisier, X. Zhuang, and P. Natarajan. Zero-shot event detection using multi-modal fusion of weakly supervised concepts. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.
- [41] Y. Xia, X. Cao, F. Wen, G. Hua, and J. Sun. Learning discriminative reconstructions for unsupervised outlier removal. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1511–1519, 2015.
- [42] Z. Xu, Y. Yang, and A. G. Hauptmann. A discriminative cnn video representation for event detection. *arXiv preprint arXiv:1411.4006*, 2014.
- [43] G. Ye, Y. Li, H. Xu, D. Liu, and S.-F. Chang. Eventnet: A large scale structured concept library for complex event detection in video. In *Proceedings of the 23rd Annual ACM Conference on Multimedia Conference*, pages 471–480. ACM, 2015.
- [44] S.-I. Yu, L. Jiang, Z. Xu, Y. Yang, and A. G. Hauptmann. Content-based video search over 1 million videos with 1 core in 1 second. In *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval*, pages 419–426. ACM, 2015.
- [45] R. Yuster and U. Zwick. Fast sparse matrix multiplication. *ACM Transactions on Algorithms (TALG)*, 1(1):2–13, 2005.